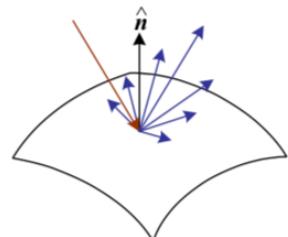


# Deep Object Recognition

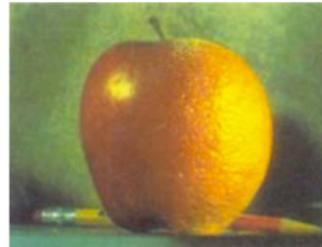
Frank Dellaert

CS 4476 Computer Vision at Georgia Tech

Several Slides by Dhruv Bathra, James Hays, Kaiming He, and others



2. Image Formation



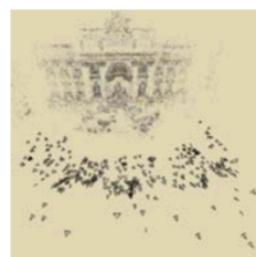
3. Image Processing



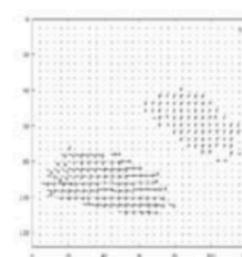
4. Features



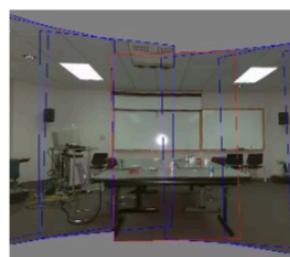
5. Segmentation



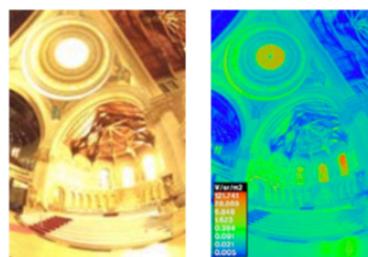
6-7. Structure from Motion



8. Motion



9. Stitching



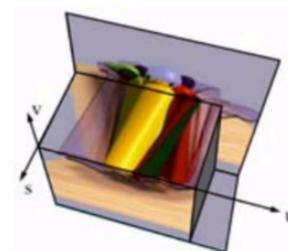
10. Computational Photography



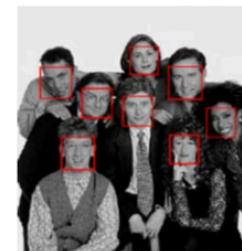
11. Stereo



12. 3D Shape

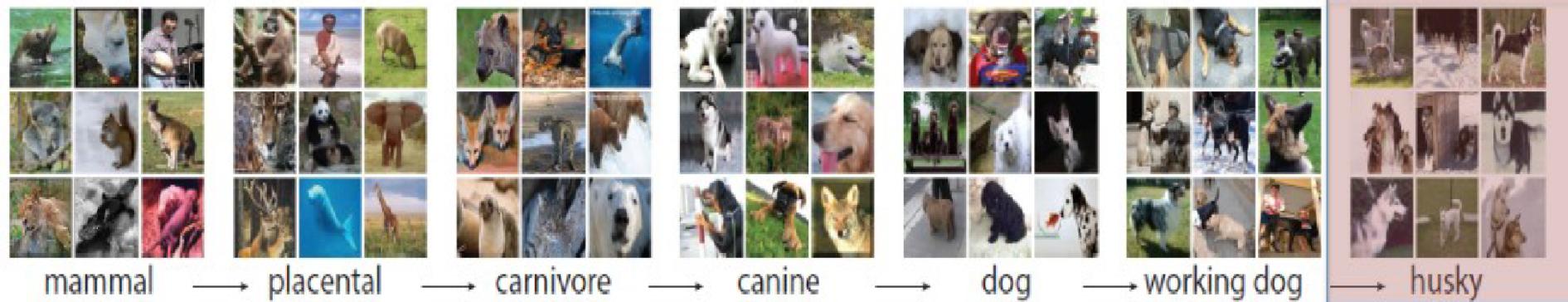


13. Image-based Rendering



14. Recognition

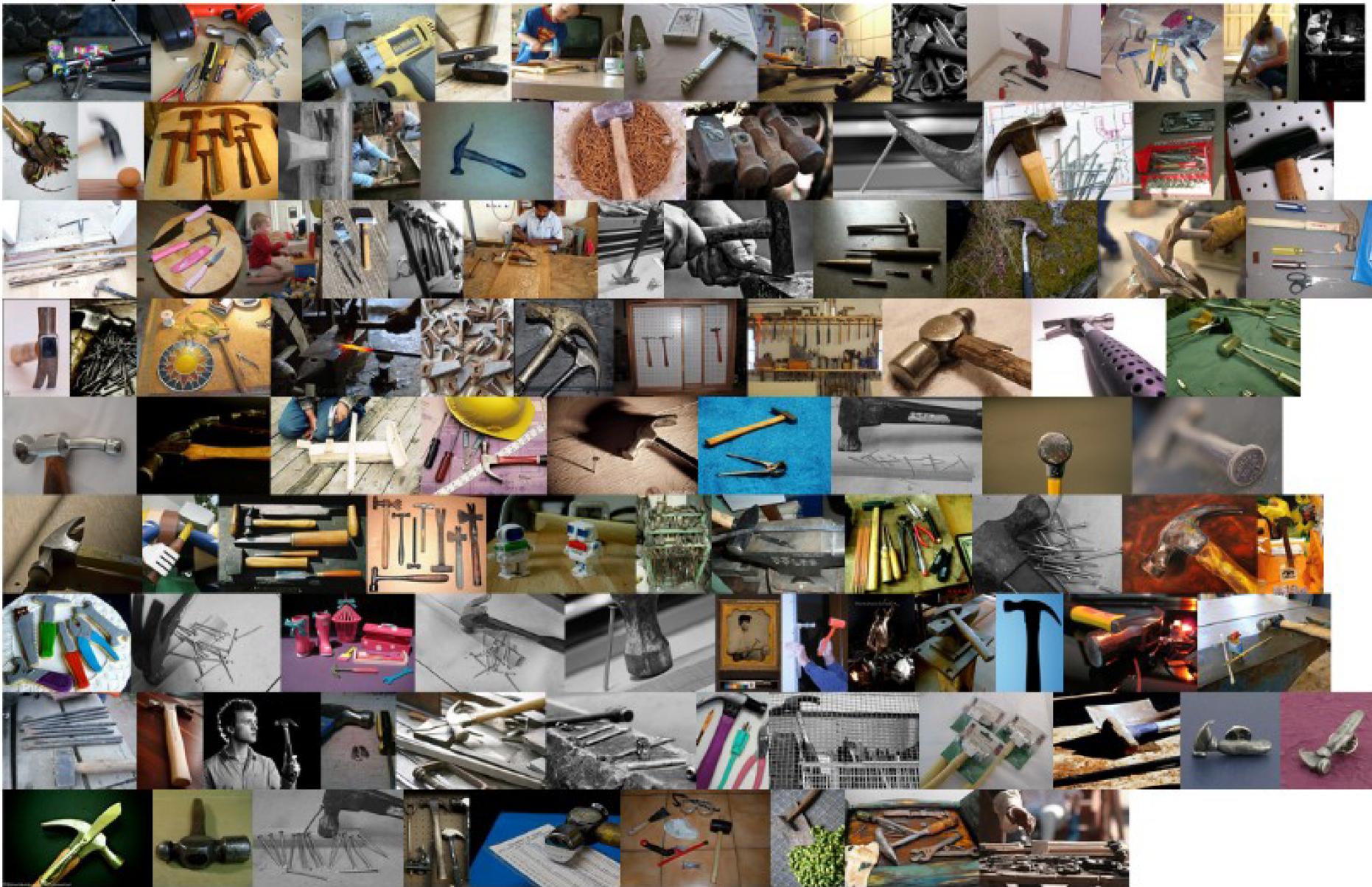
# Dataset: ImageNet 2012



- [S: \(n\) Eskimo dog](#), husky (breed of heavy-coated Arctic sled dog)
  - *direct hypernym / inherited hypernym / sister term*
  - [S: \(n\) working dog](#) (any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs)
  - [S: \(n\) dog, domestic dog, \*Canis familiaris\*](#) (a member of the genus *Canis* (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) "the dog barked all night"
  - [S: \(n\) canine, canid](#) (any of various fissiped mammals with non-retractile claws and typically long muzzles)
  - [S: \(n\) carnivore](#) (a terrestrial or aquatic flesh-eating mammal) "terrestrial carnivores have four or five clawed digits on each limb"
  - [S: \(n\) placental, placental mammal, eutherian, eutherian mammal](#) (mammals having a placenta; all mammals except monotremes and marsupials)
  - [S: \(n\) mammal, mammalian](#) (any warm-blooded vertebrate having the skin more or less covered with hair; young are born alive except for the small subclass of monotremes and nourished with milk)
  - [S: \(n\) vertebrate, craniate](#) (animals having a bony or cartilaginous skeleton with a segmented spinal column and a large brain enclosed in a skull or cranium)
  - [S: \(n\) chordate](#) (any animal of the phylum Chordata having a notochord or spinal column)
  - [S: \(n\) animal, animate being, beast, brute, creature, fauna](#) (a living organism characterized by voluntary movement)
  - [S: \(n\) organism, being](#) (a living thing that has (or can develop) the ability to act or function independently)
  - [S: \(n\) living thing, animate thing](#) (a living (or once living) entity)
  - [S: \(n\) whole, unit](#) (an assemblage of parts that is regarded as a single entity) "how big is that part compared to the whole?"; "the team is a unit"
  - [S: \(n\) object, physical object](#) (a tangible and visible entity, an entity that can cast a shadow) "it was full of rackets, balls and other objects"
  - [S: \(n\) physical entity](#) (an entity that has physical existence)
  - [S: \(n\) entity](#) (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

# ImageNet

Examples of hammer:



# AlexNet

- CNN by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton
- Competed in the ImageNet Large Scale Visual Recognition Challenge on September 30, 2012. Achieved a top-5 error of 15.3%, beating SOTA by 10%.
- Seen by many as the start of the DL revolution in CV.
- That claim is contested by Jürgen Schmidhuber, whose postdoc Dan Ciresan published a similar result in IJCAI 2011 (but on easier datasets).
- Both owe a debt to Fukushima, who invented CNNs in 1980, and Yann LeCun, who applied backprop to CNNs in 89.

---

## ImageNet Classification with Deep Convolutional Neural Networks

---

Alex Krizhevsky  
University of Toronto  
kriz@cs.utoronto.ca

Ilya Sutskever  
University of Toronto  
ilya@cs.utoronto.ca

Geoffrey E. Hinton  
University of Toronto  
hinton@cs.utoronto.ca

### Abstract

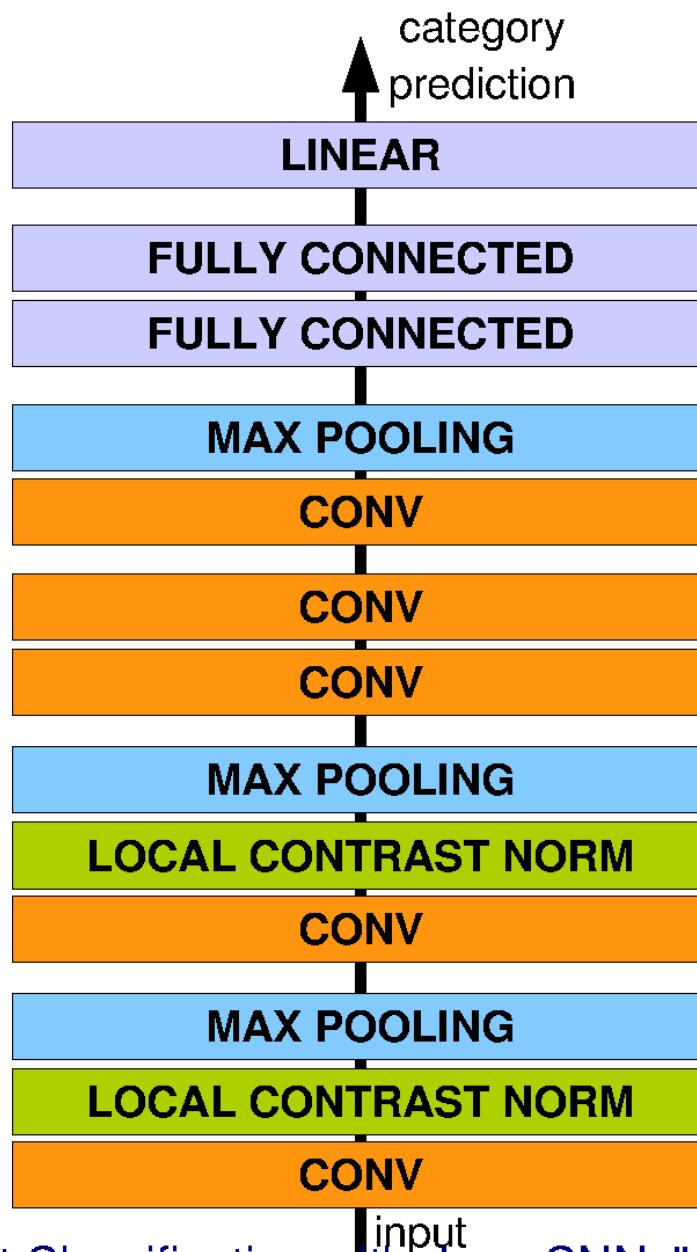
We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called “dropout” that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

### 1 Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-10/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance [4]. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Pinto et al. [21]), but it has only recently become possible to collect labeled datasets with millions of images. The new larger datasets include LabelMe [23], which consists of hundreds of thousands of fully-segmented images, and ImageNet [6], which consists of over 15 million labeled high-resolution images in over 22,000 categories.

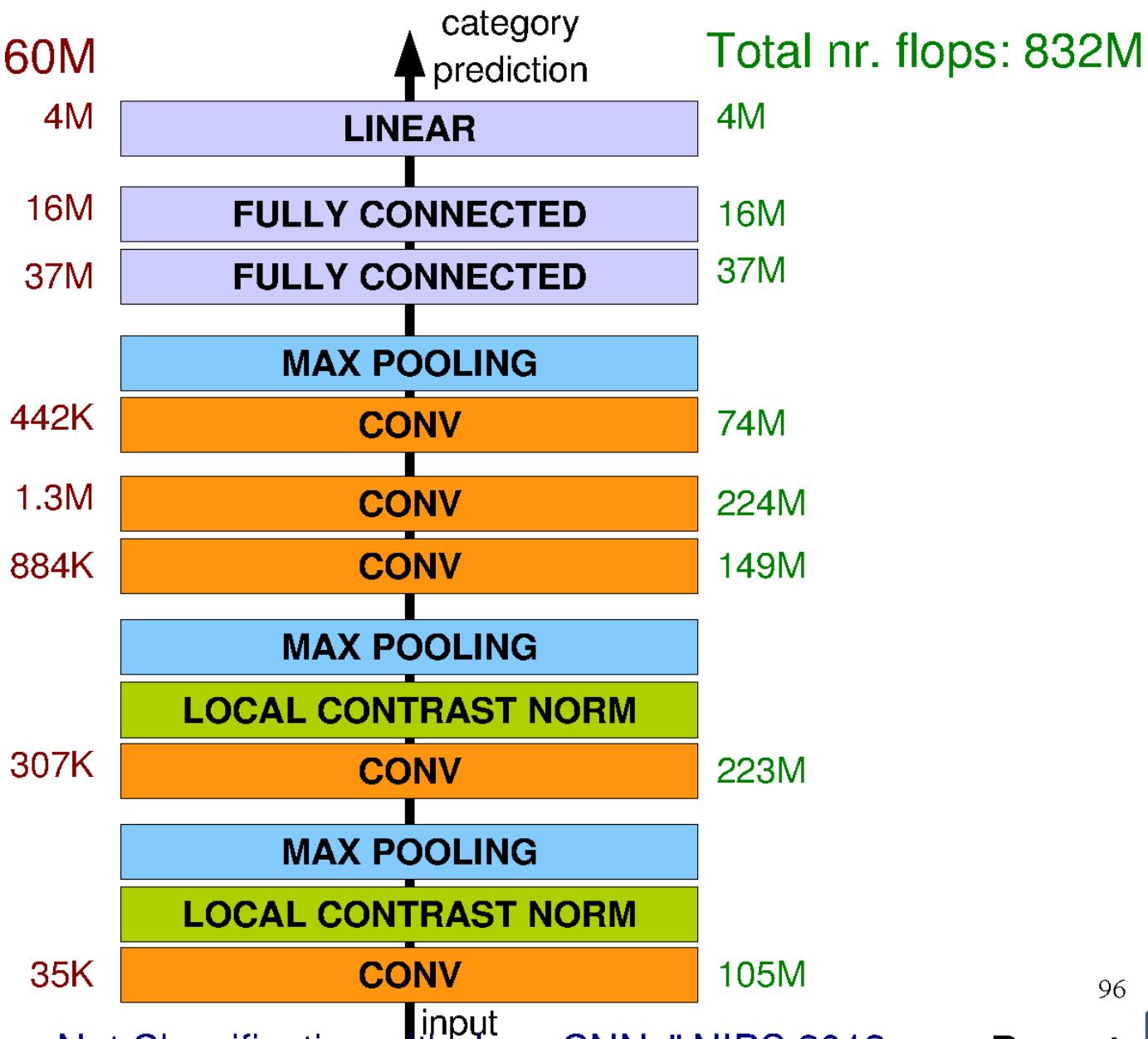
To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we don’t have. Convolutional neural networks (CNNs) constitute one such class of models [16, 11, 13, 18, 15, 22, 26]. Their capacity can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images (namely, stationarity of statistics and locality of pixel dependencies). Thus, compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer connections and parameters and so they are easier to train, while their theoretically-best performance is likely to be only slightly worse.

# Architecture for Classification



# Architecture for Classification

Total nr. params: 60M



# Optimization

## SGD with momentum:

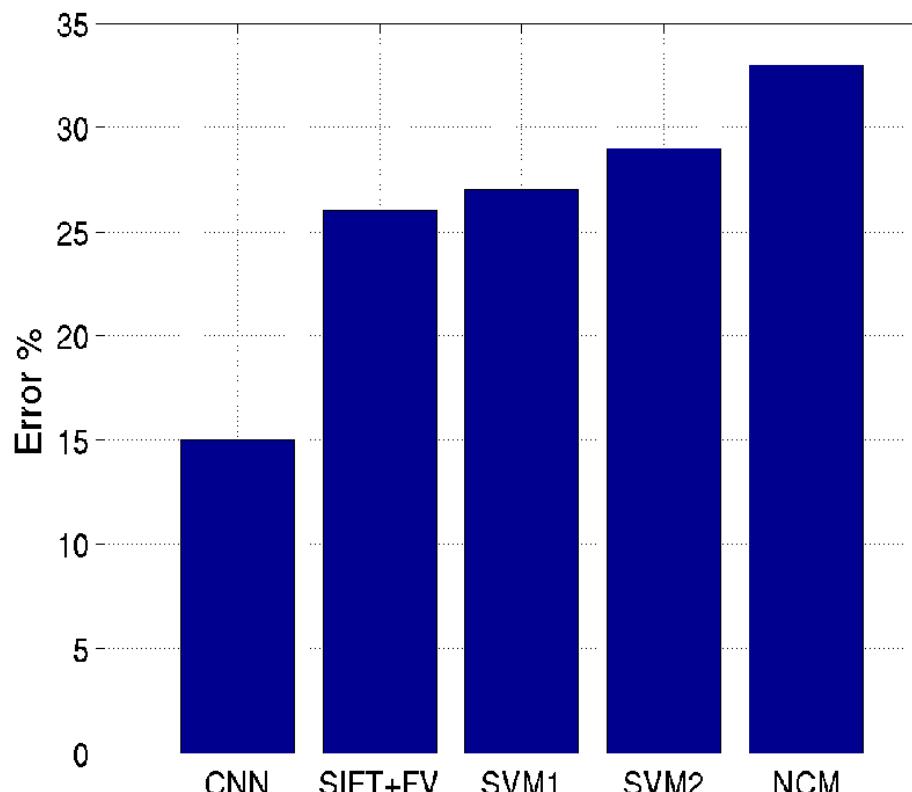
- Learning rate = 0.01
- Momentum = 0.9

## Improving generalization by:

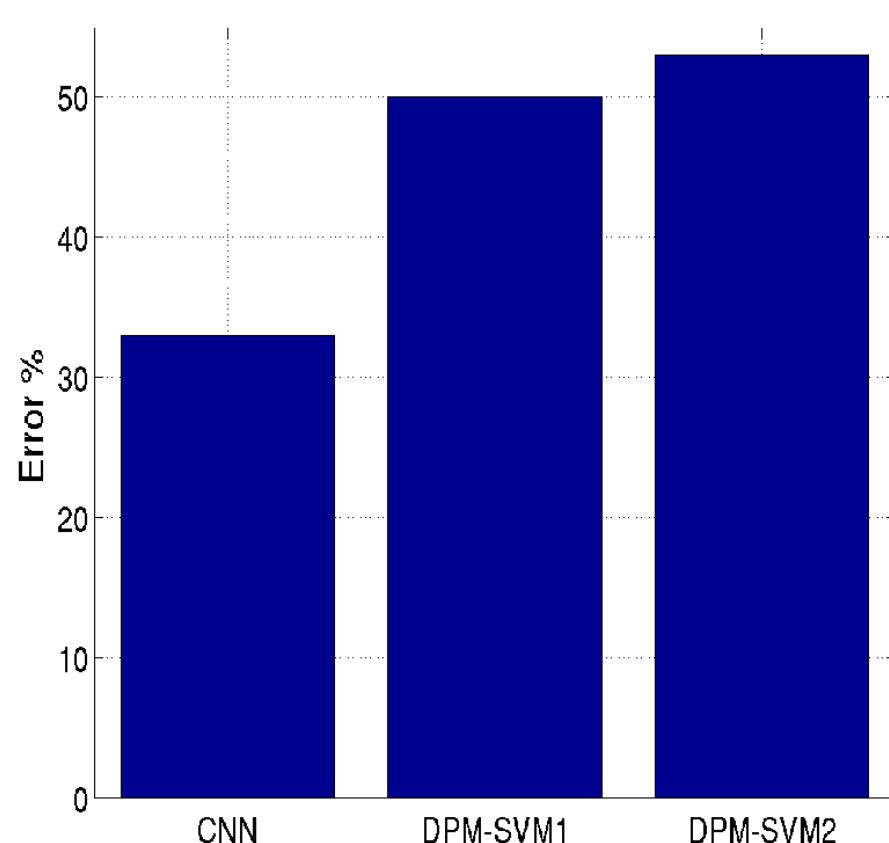
- Weight sharing (convolution)
- Input distortions
- Dropout = 0.5
- Weight decay = 0.0005

# Results: ILSVRC 2012

TASK 1 - CLASSIFICATION



TASK 2 - DETECTION





mite	mite	container ship	motor scooter	leopard
black widow		lifeboat	motor scooter	leopard
cockroach		amphibian	go-kart	jaguar
tick		fireboat	moped	cheetah
starfish		drilling platform	bumper car	snow leopard
			golfcart	Egyptian cat



convertible	agaric	dalmatian	squirrel monkey
grille	mushroom	grape	spider monkey
pickup	jelly fungus	elderberry	titi
beach wagon	gill fungus	ffordshire bullterrier	indri
fire engine	dead-man's-fingers	currant	howler monkey

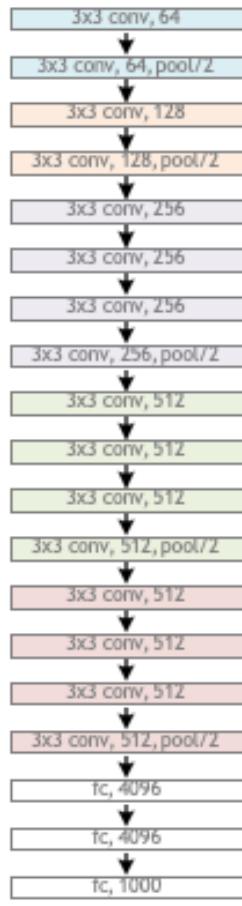
# Beyond AlexNet

# VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

**Karen Simonyan & Andrew Zisserman 2015**

**These are the “VGG” networks.**

# VGG



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 <b>LRN</b>	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 <b>conv3-256</b> <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 <b>conv3-512</b> <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 <b>conv3-512</b> <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: Number of parameters (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Table 4: ConvNet performance at multiple test scales.

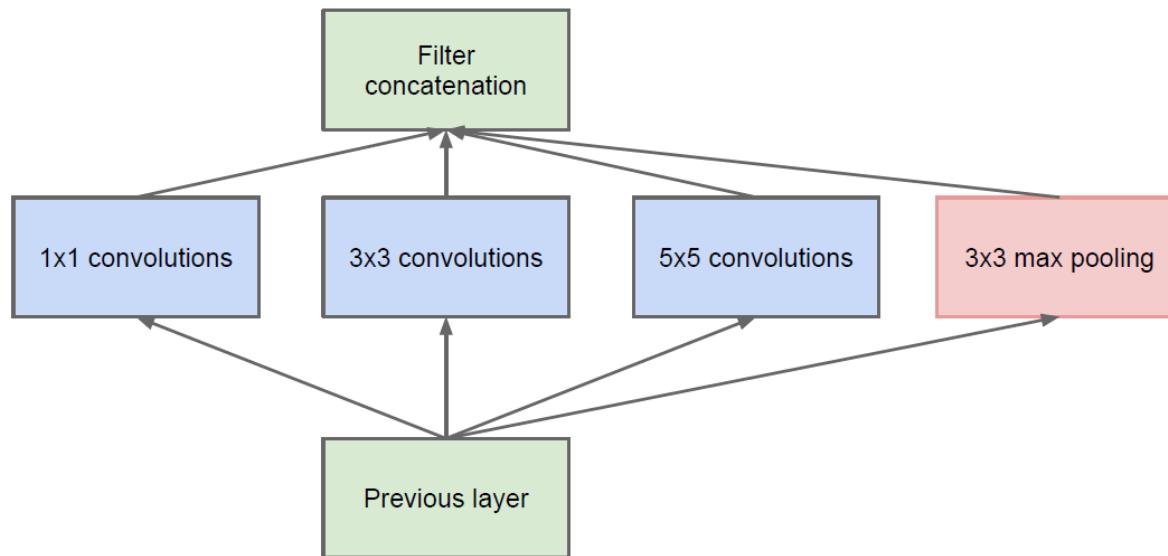
ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train ( $S$ )	test ( $Q$ )		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	<b>24.8</b>	<b>7.5</b>
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	<b>24.8</b>	<b>7.5</b>

# Going Deeper with Convolutions

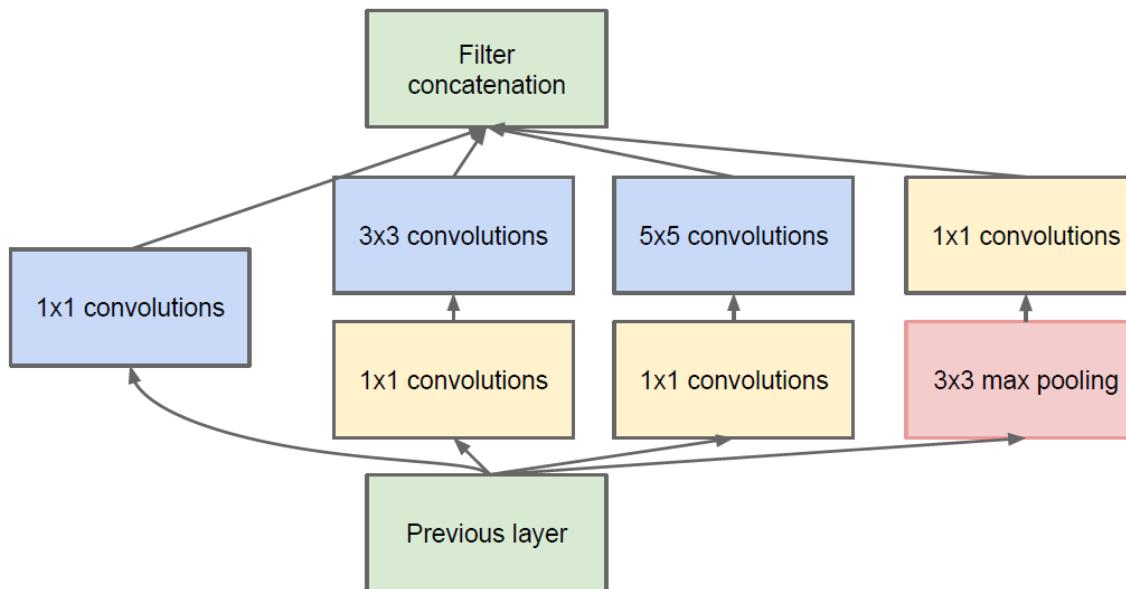
**Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed,  
Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich  
2015**

**This is the “Inception” architecture or “GoogLeNet”**

**\*The architecture blocks are called “Inception” modules  
and the collection of them into a particular net is “GoogLeNet”**

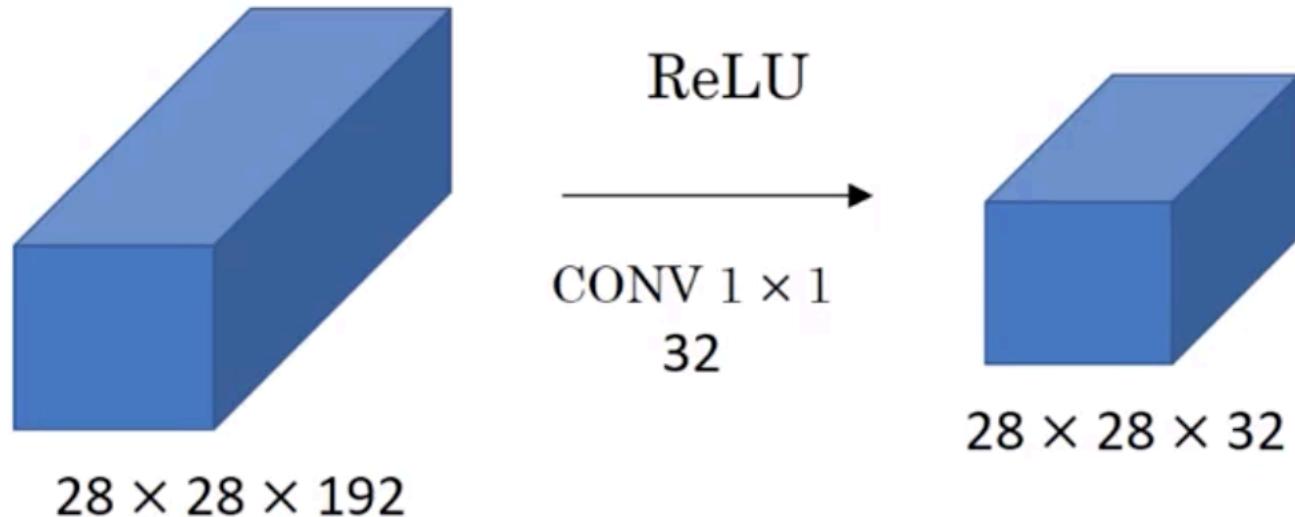


(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

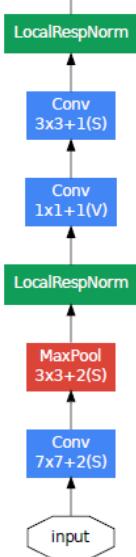
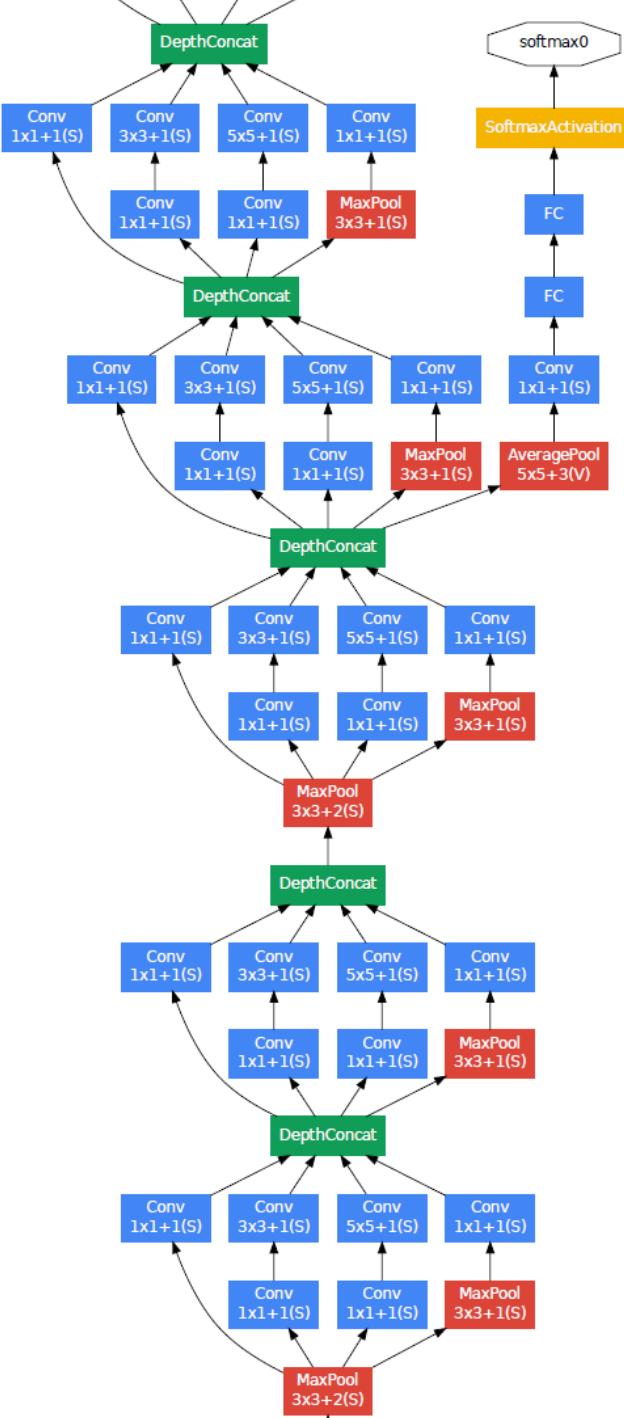
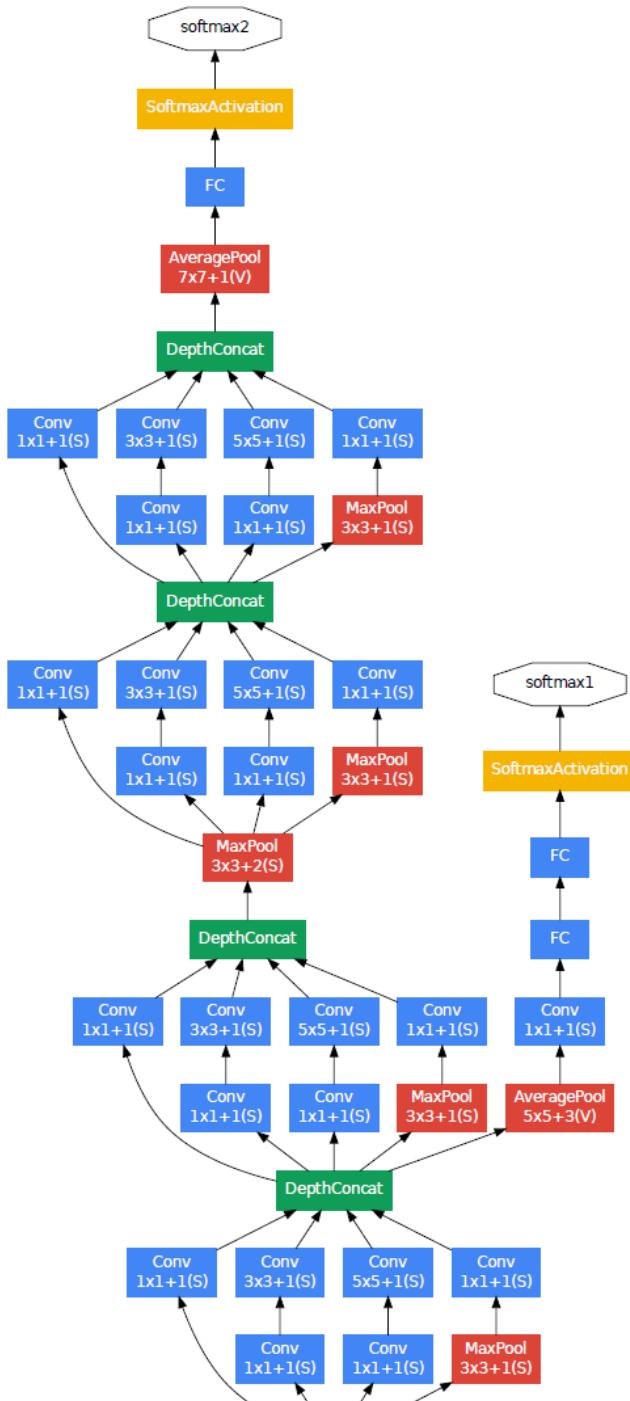
# 1x1 Convolutions



- Linearly reduced a set of  $n$  features to a set of  $m$  features. Example:  $192 \rightarrow 32$
- I.e., matrix multiplication with  $m \times n$  matrix, at each location
- Typically followed by ReLU

<b>type</b>	<b>patch size/ stride</b>	<b>output size</b>	<b>depth</b>	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

GoogLeNet: Only 6.8 million parameters. AlexNet ~60 million, VGG up to 138 million



# Results

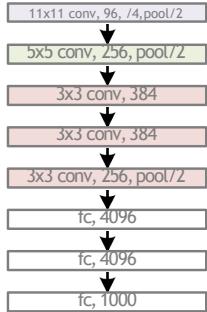
- ILSVRC 2014:

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

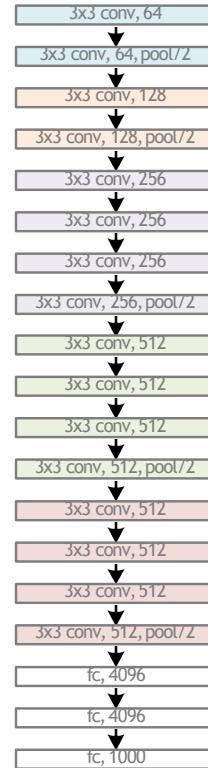
Table 2: Classification performance.

# Revolution of Depth

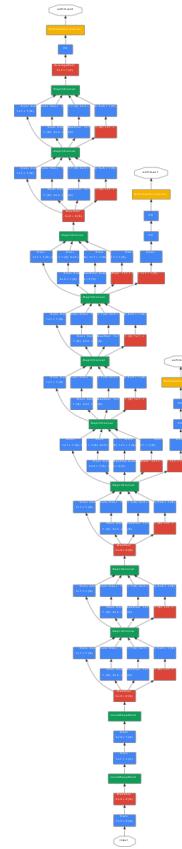
AlexNet, 8 layers  
(ILSVRC 2012)



VGG, 19 layers  
(ILSVRC 2014)



GoogleNet, 22 layers  
(ILSVRC 2014)



Surely it would be ridiculous to go any deeper...

## Introducing: ResNet

AlexNet, 8 layers  
(ILSVRC 2012)



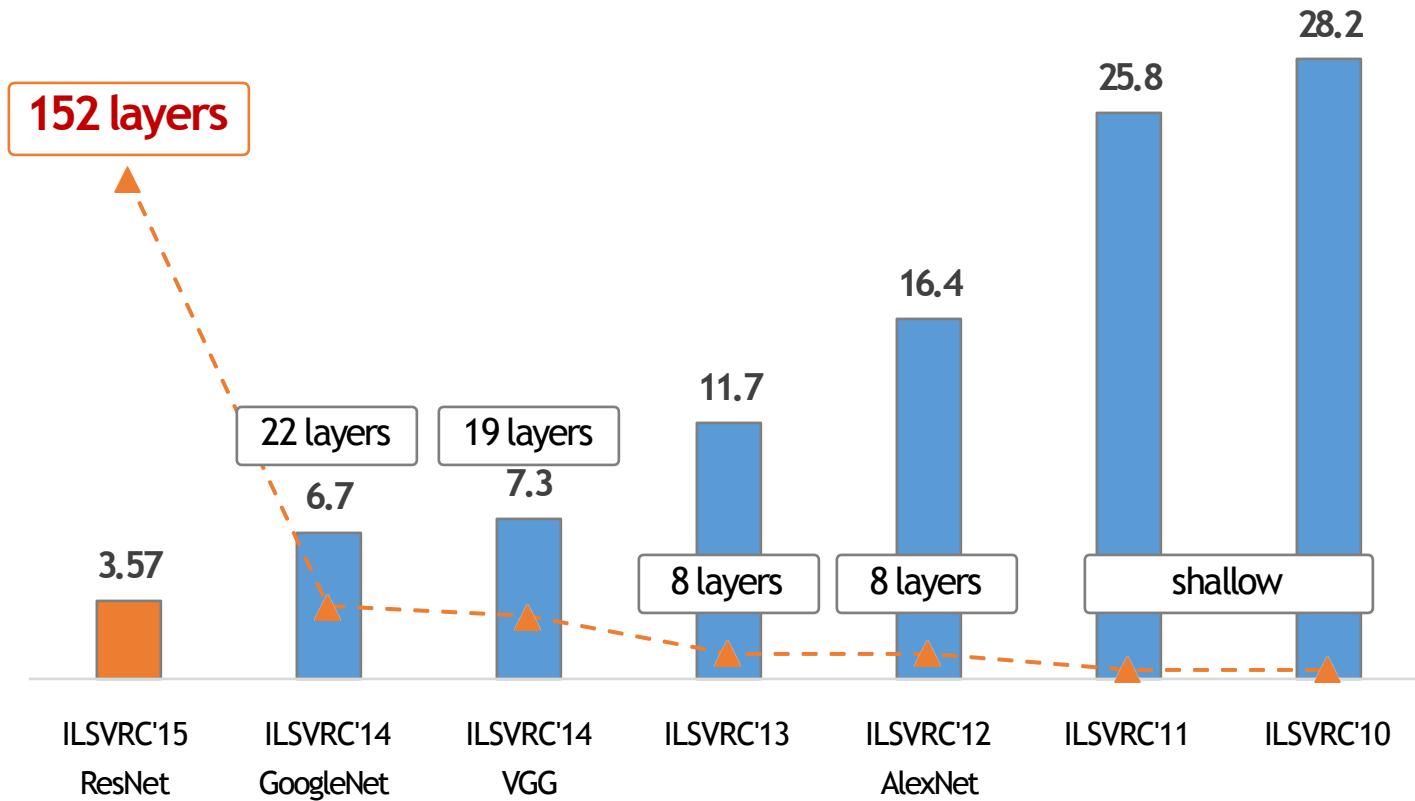
VGG, 19 layers  
(ILSVRC 2014)



ResNet, **152 layers**  
(ILSVRC 2015)

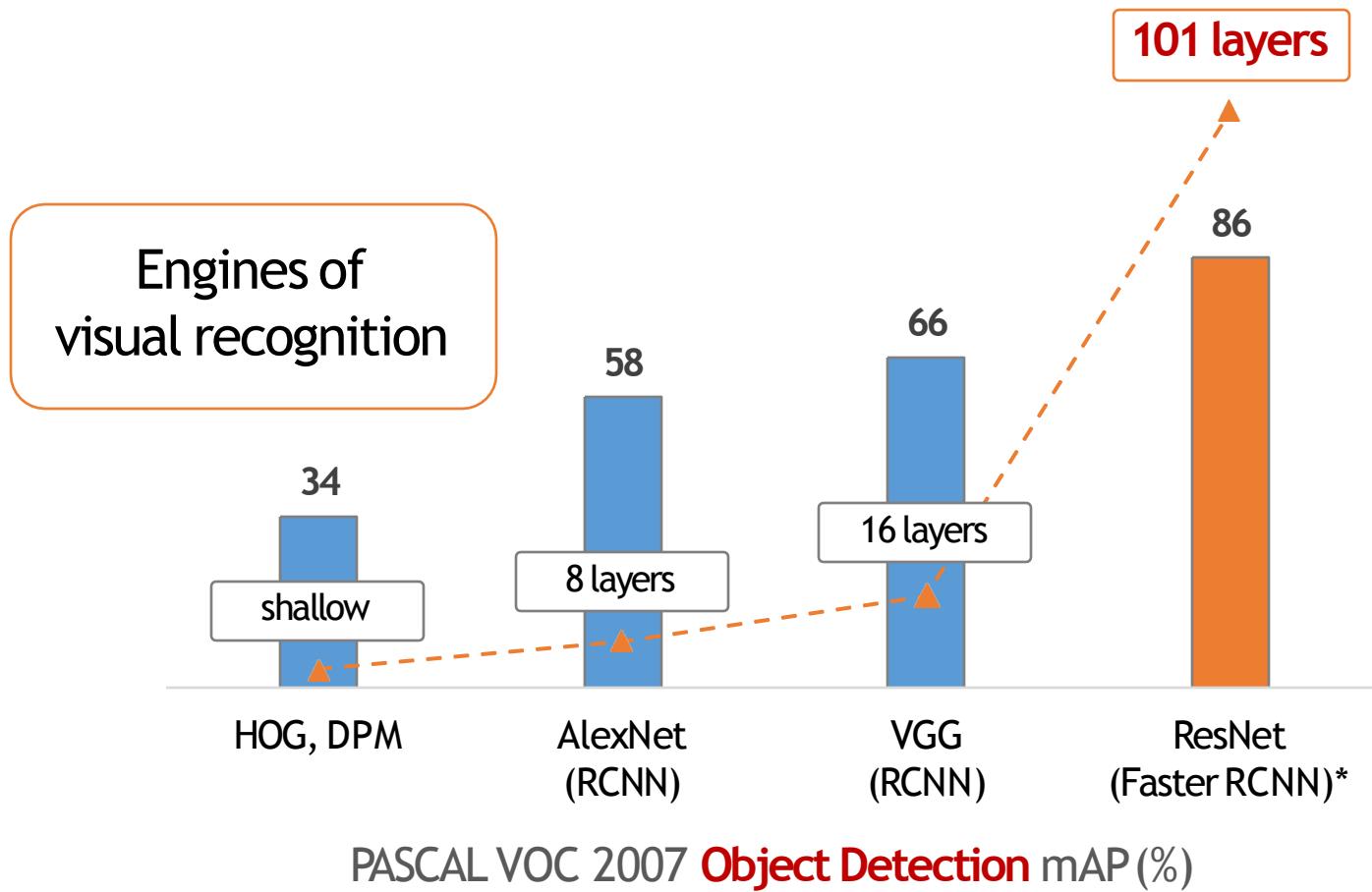


# Revolution of Depth



ImageNet Classification top-5 error (%)

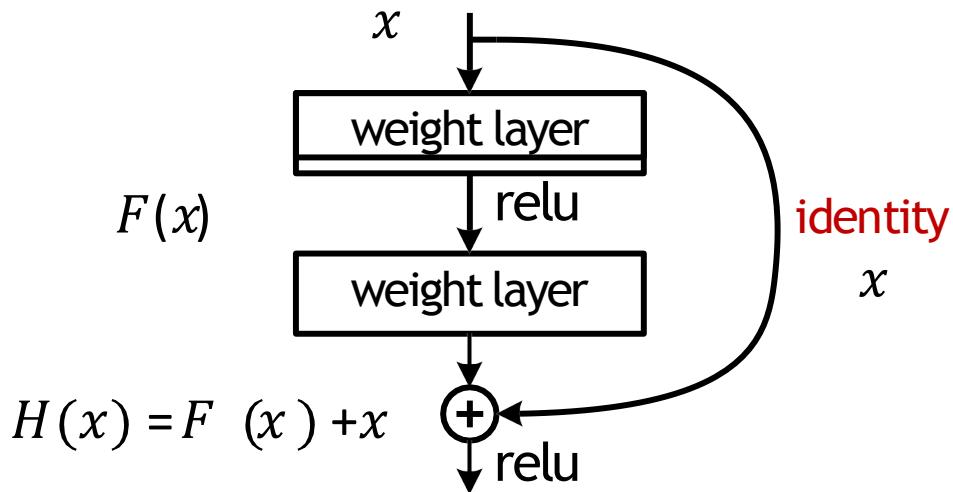
# Revolution of Depth



\*w/ other improvements & more data

# Deep Residual Learning

- $F(x)$  is a **residual mapping w.r.t. identity**

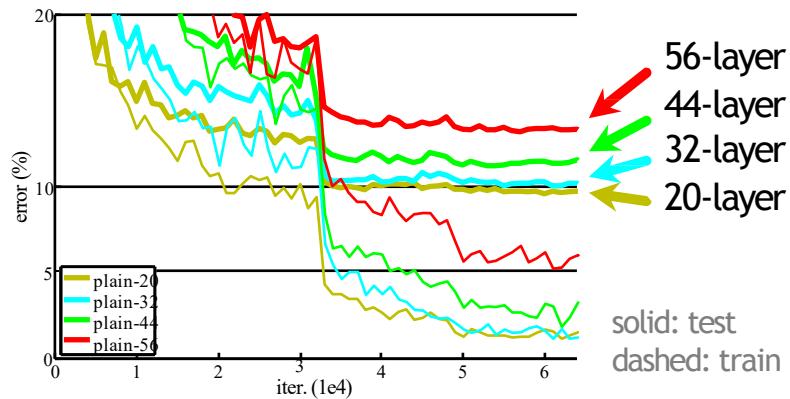


- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

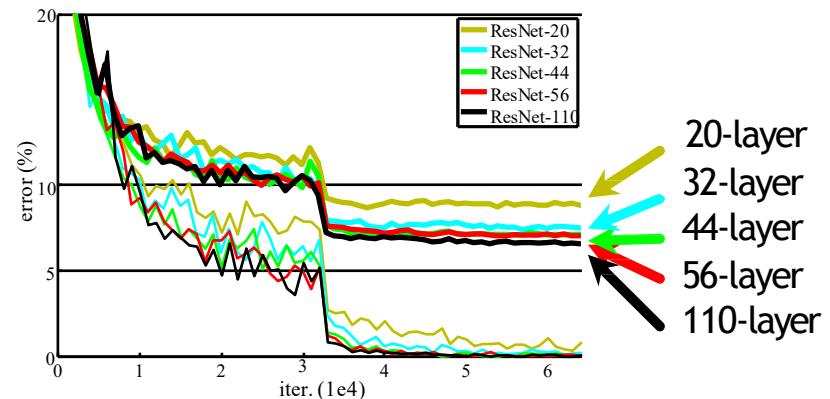
# CIFAR-10

## experiments

CIFAR-10 plain nets



CIFAR-10 ResNets



- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

# ResNets @ ILSVRC & COCO 2015 Competitions

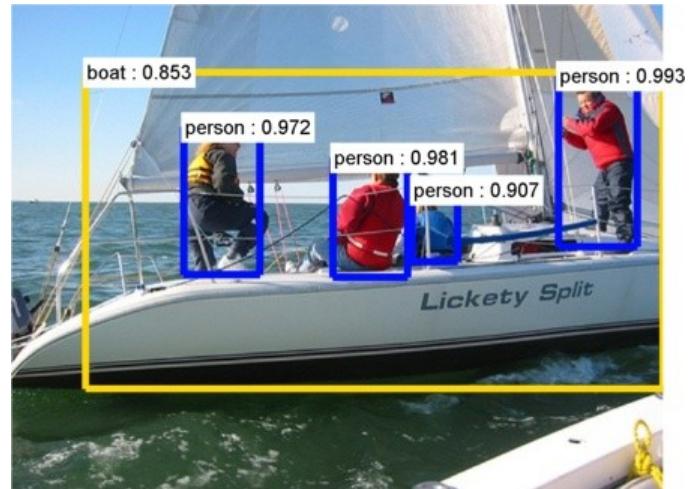
- **1st places in all five main tracks**
  - ImageNet Classification: “*Ultra-deep*” **152-layer** nets
  - ImageNet Detection: **16%** better than 2nd
  - ImageNet Localization: **27%** better than 2nd
  - COCO Detection: **11%** better than 2nd
  - COCO Segmentation: **12%** better than 2nd
- **33K citations (in 4 years)**

\*improvements are relative numbers

# Example: Object Detection



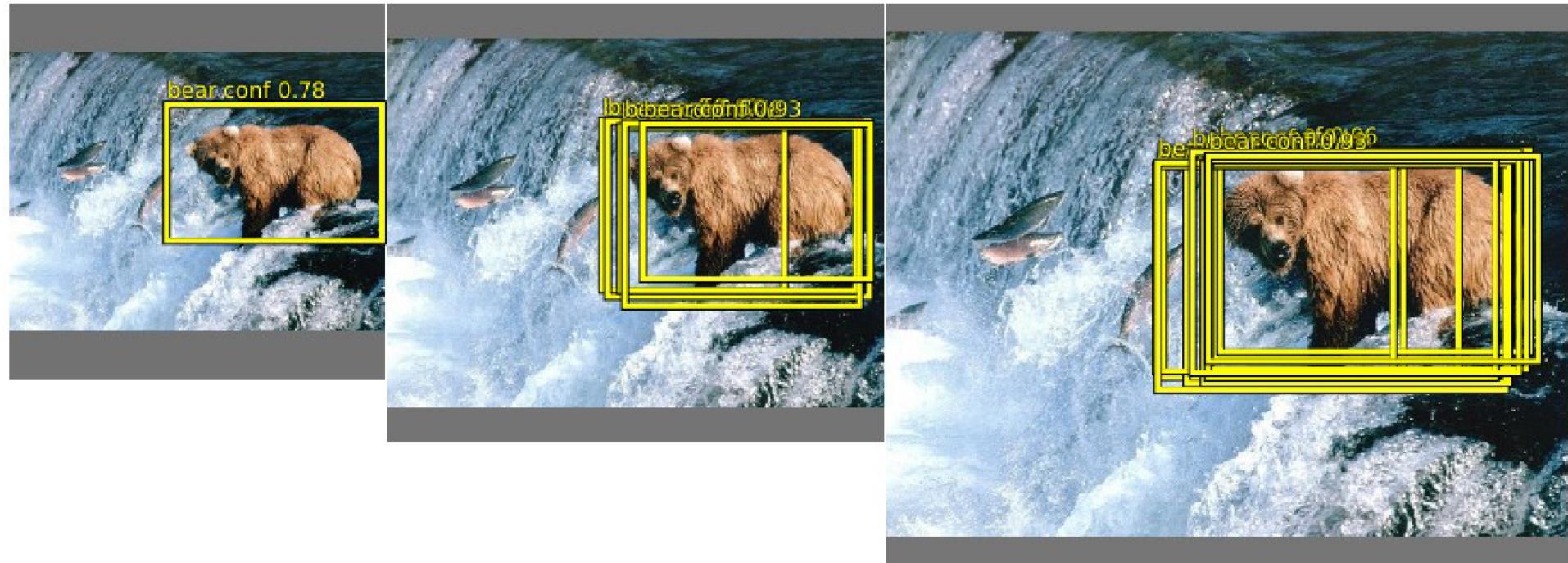
Image Classification  
(what?)



Object Detection  
(what + where?)

# CONV NETS: EXAMPLES

## - Object detection



Sermanet et al. “OverFeat: Integrated recognition, localization, ...” arxiv 2013

Girshick et al. “Rich feature hierarchies for accurate object detection...” arxiv 2013 91

Szegedy et al. “DNN for object detection” NIPS 2013

# Object Detection: R-CNN

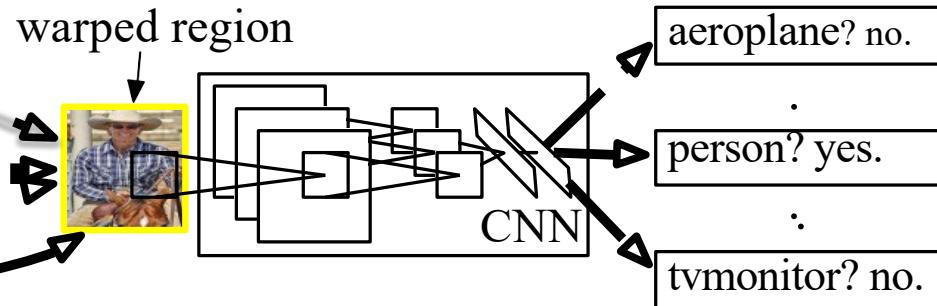
figure credit: R. Girshick et al.



input image



region proposals  
~2,000



1 CNN for each region  
-> 4096 feature vector

classify regions

## Region-based CNN pipeline

Proposals by “Selective search” algorithm (2013)

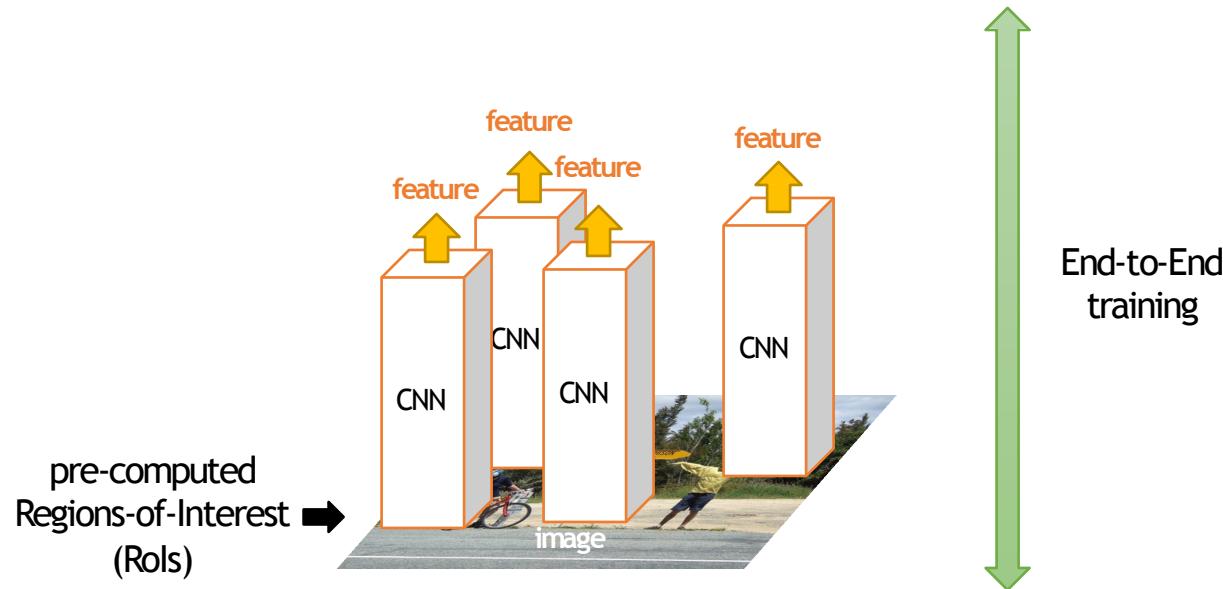
Two “heads”:

- classifier
- BB regressor

Nice post: <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>

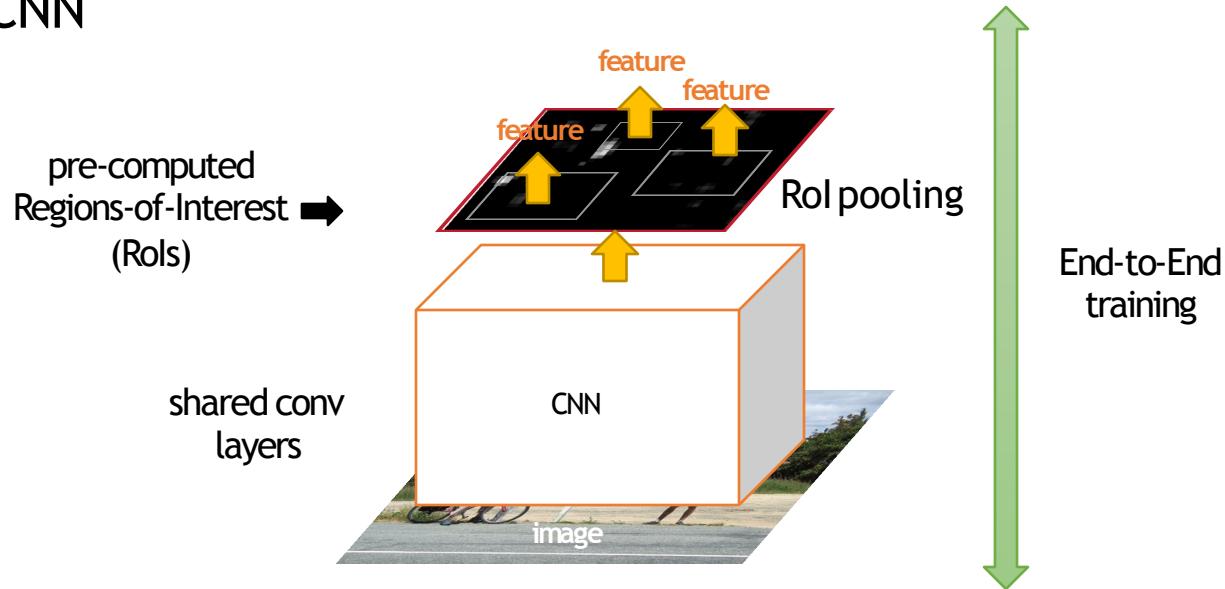
# Object Detection: R-CNN

- R-CNN



# Object Detection: Fast R-CNN

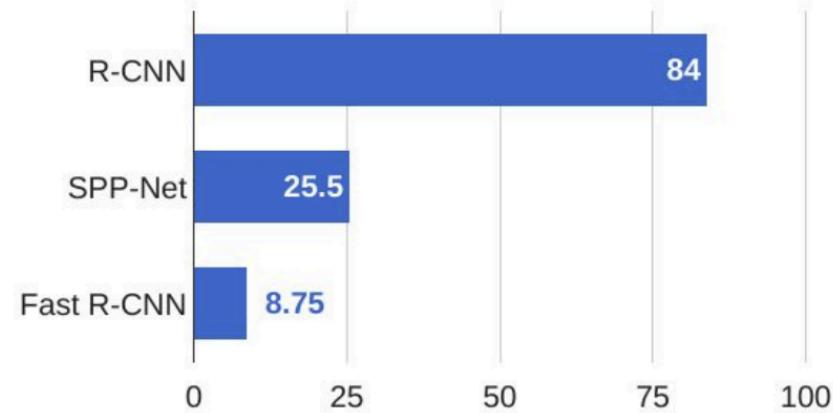
- Fast R-CNN



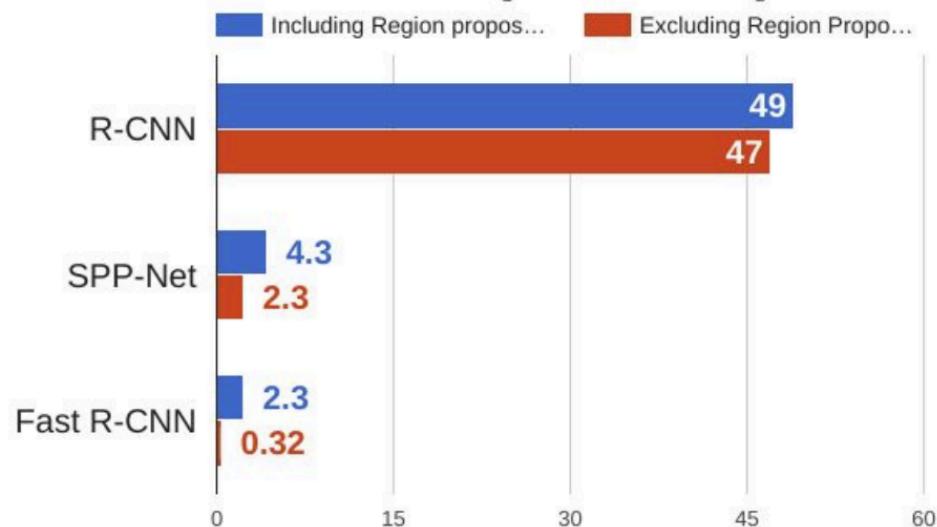
# Object Detection: Fast R-CNN

- Fast R-CNN

**Training time (Hours)**

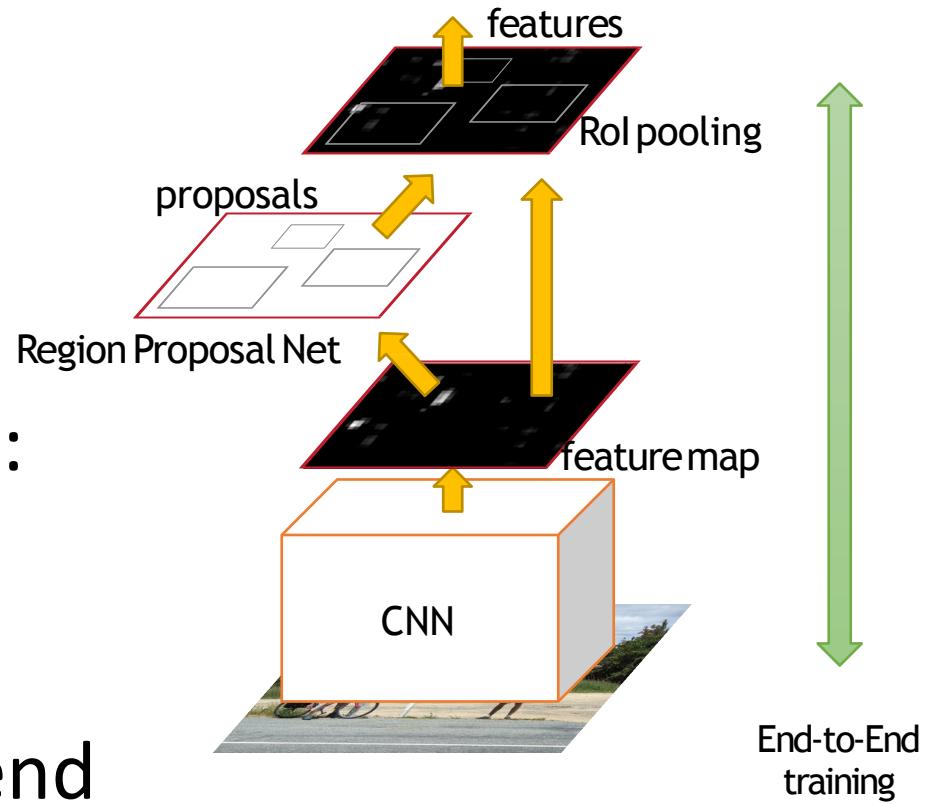


**Test time (seconds)**

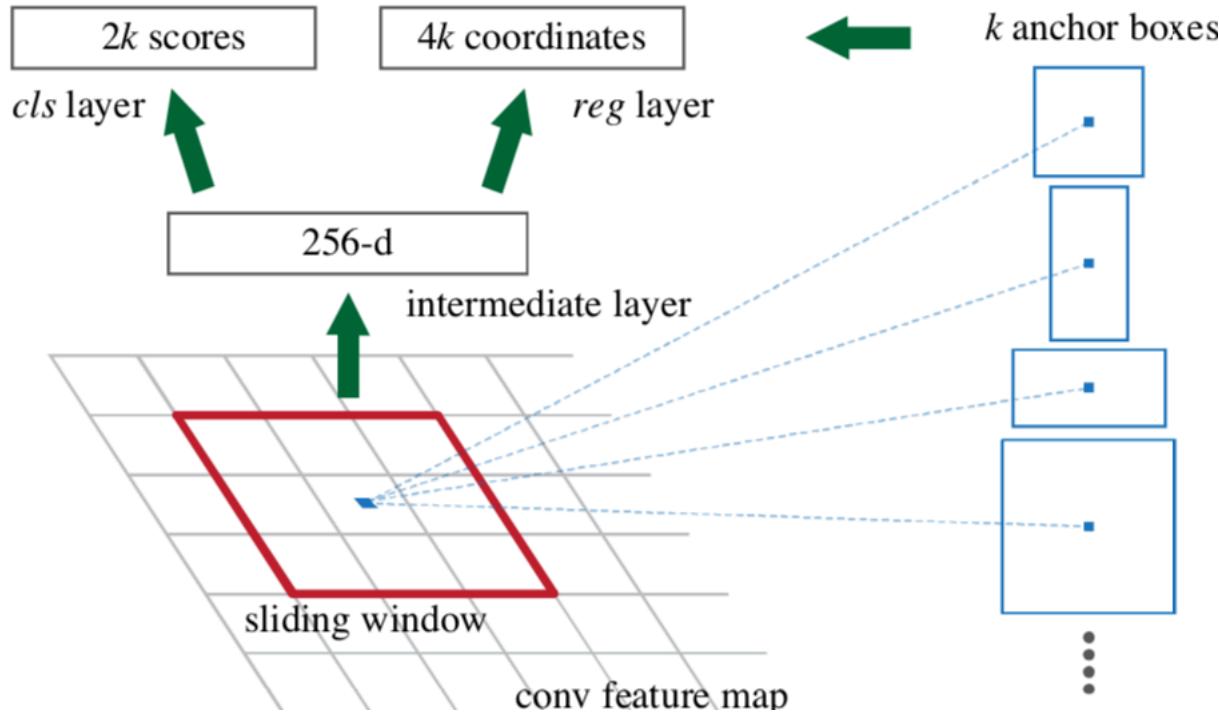


# Object Detection: Faster R-CNN

- Introduces “Region Proposal Networks” (RPNs)
- Solely based on CNN: use for classification *and* regions
- Each step is end-to-end

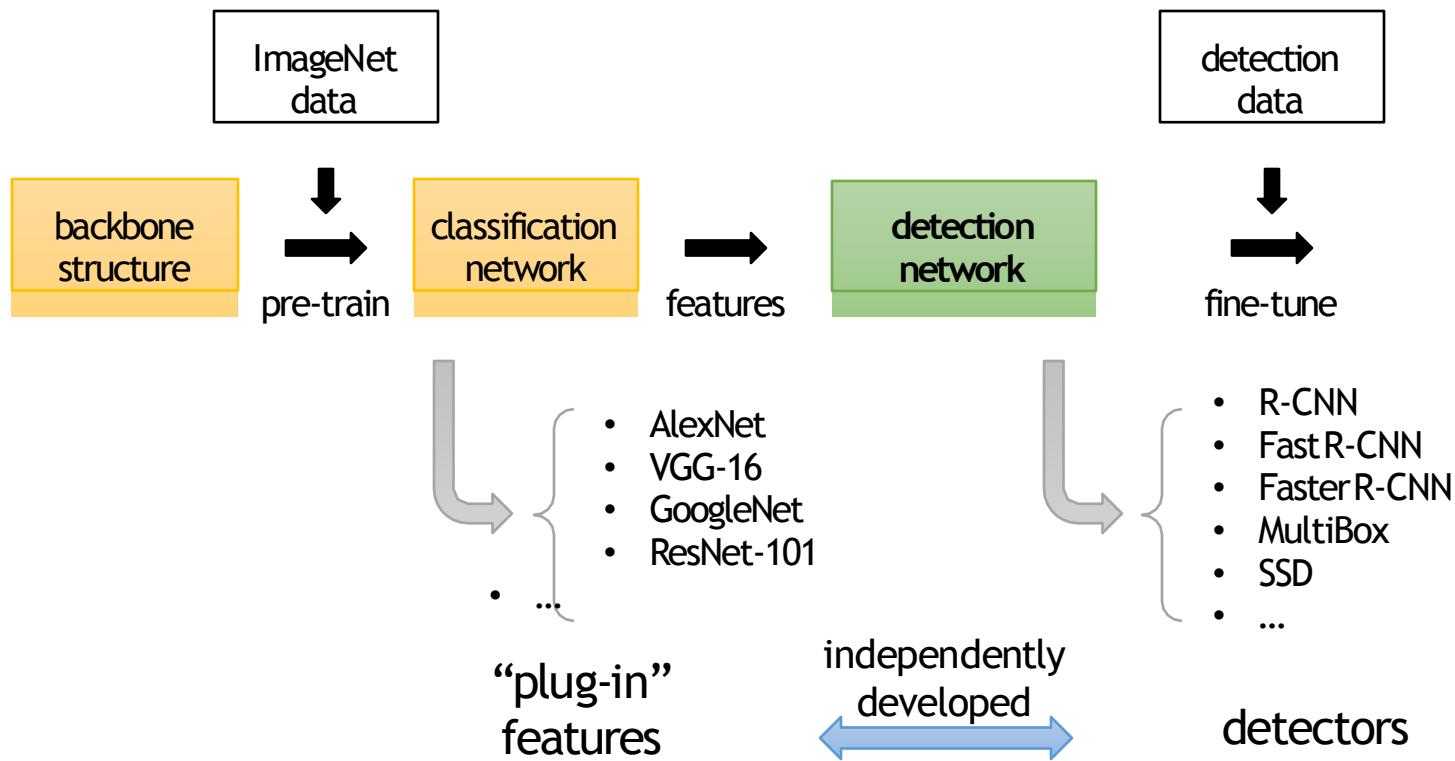


# Region Proposal Nets in Faster R-CNN



- In paper:  $k=9$  (3 scales, 3 aspect ratios)
- Sibling objectness (2k) and BB regression(4k) outputs

# Object Detection

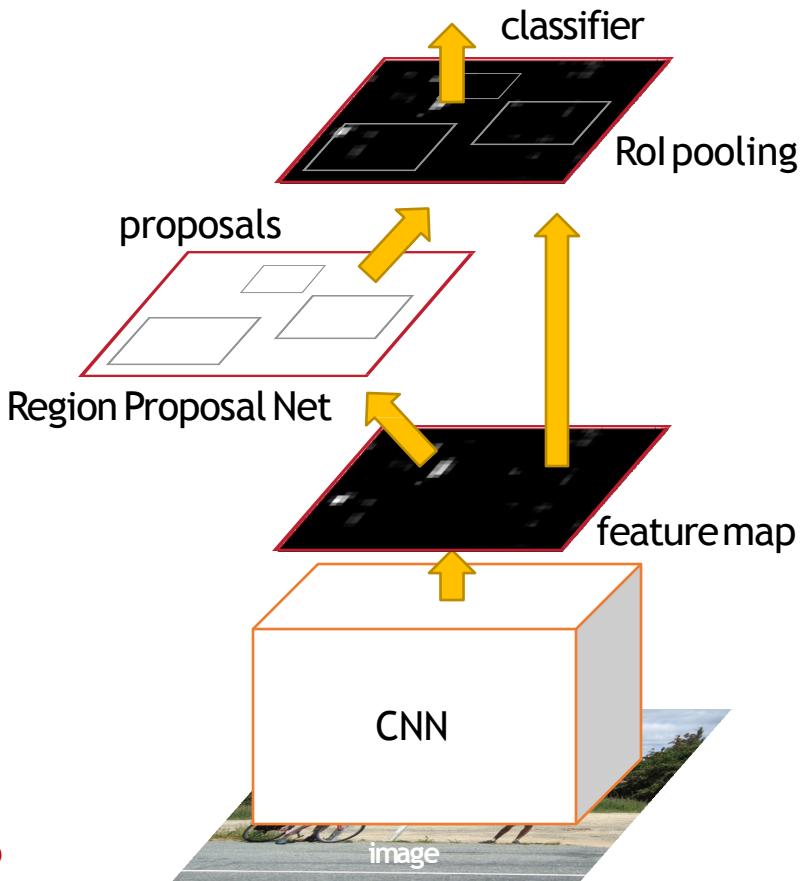


# Faster R-CNN w Resnet

- Simply “Faster R-CNN + ResNet”

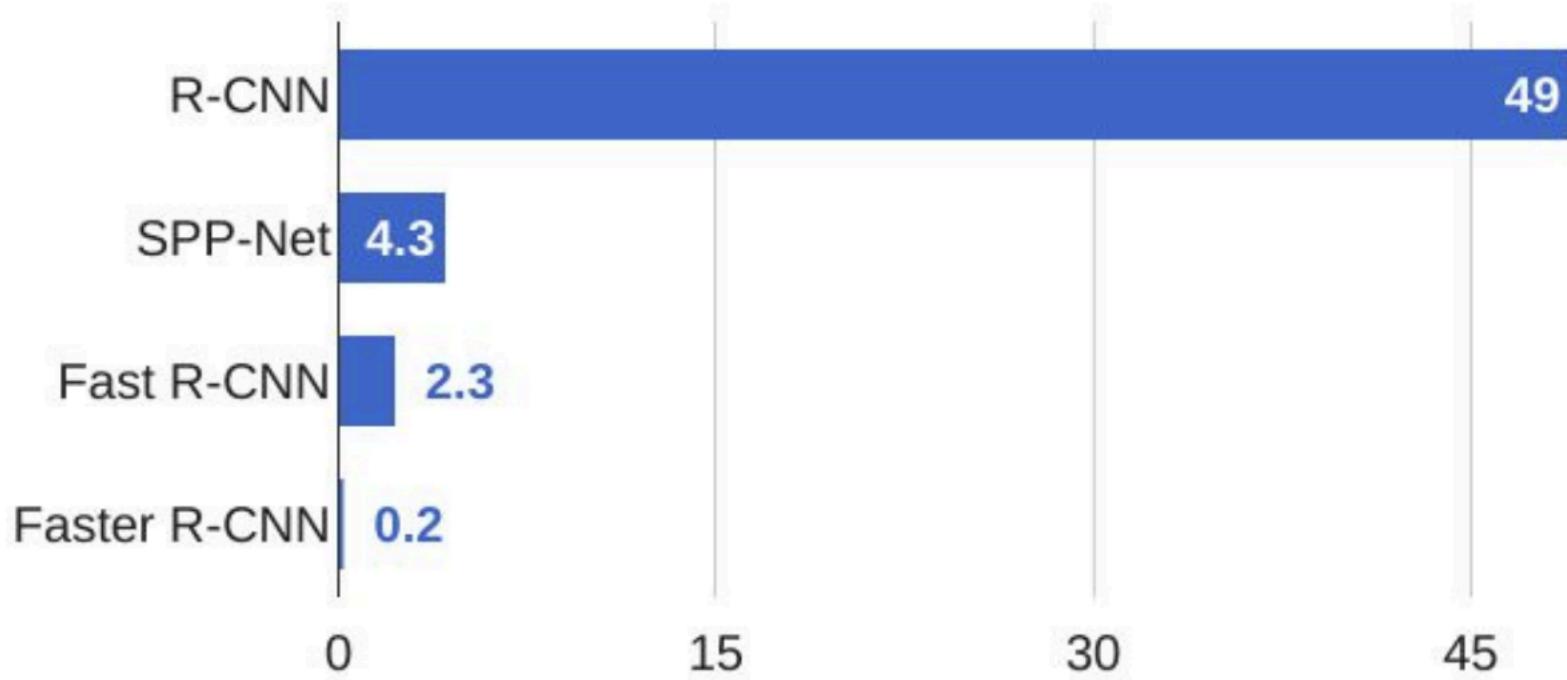
CNN	mAP@.5	mAP@.5:.95
VGG-16	41.5	21.5
ResNet-101	<b>48.4</b>	<b>27.2</b>

COCO detection results  
**ResNet-101 has 28% relative gain vs VGG-16**



# Faster R-CNN Efficiency

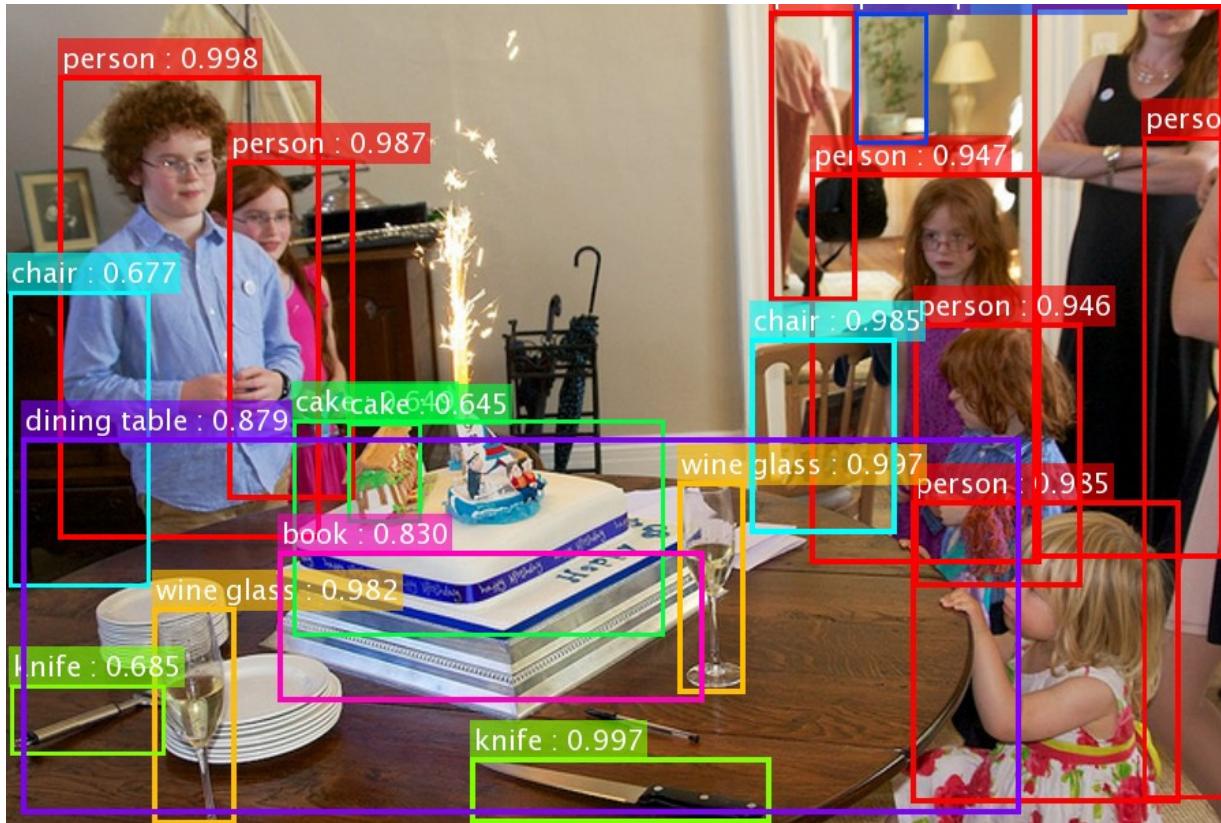
**R-CNN Test-Time Speed**



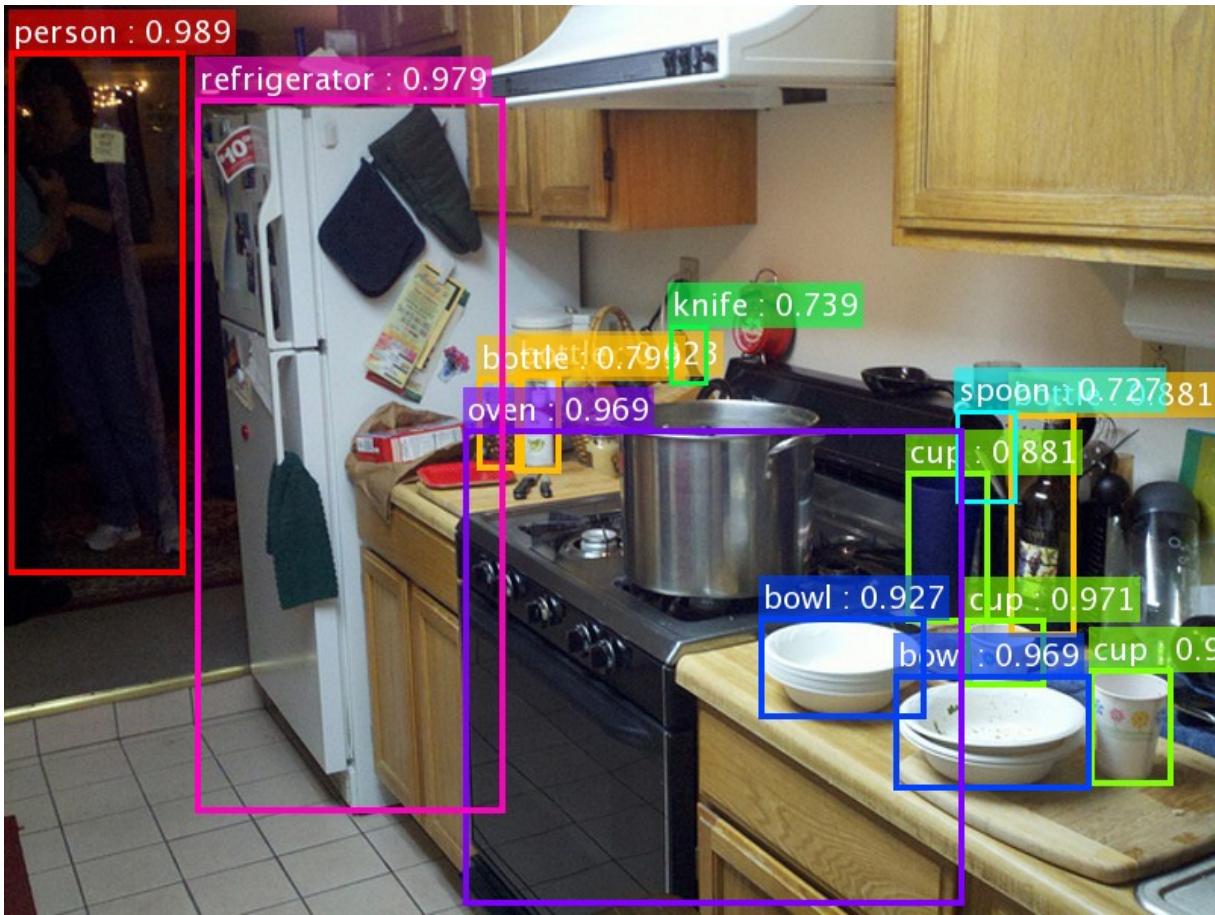
- Expensive “Selective Search” is gone

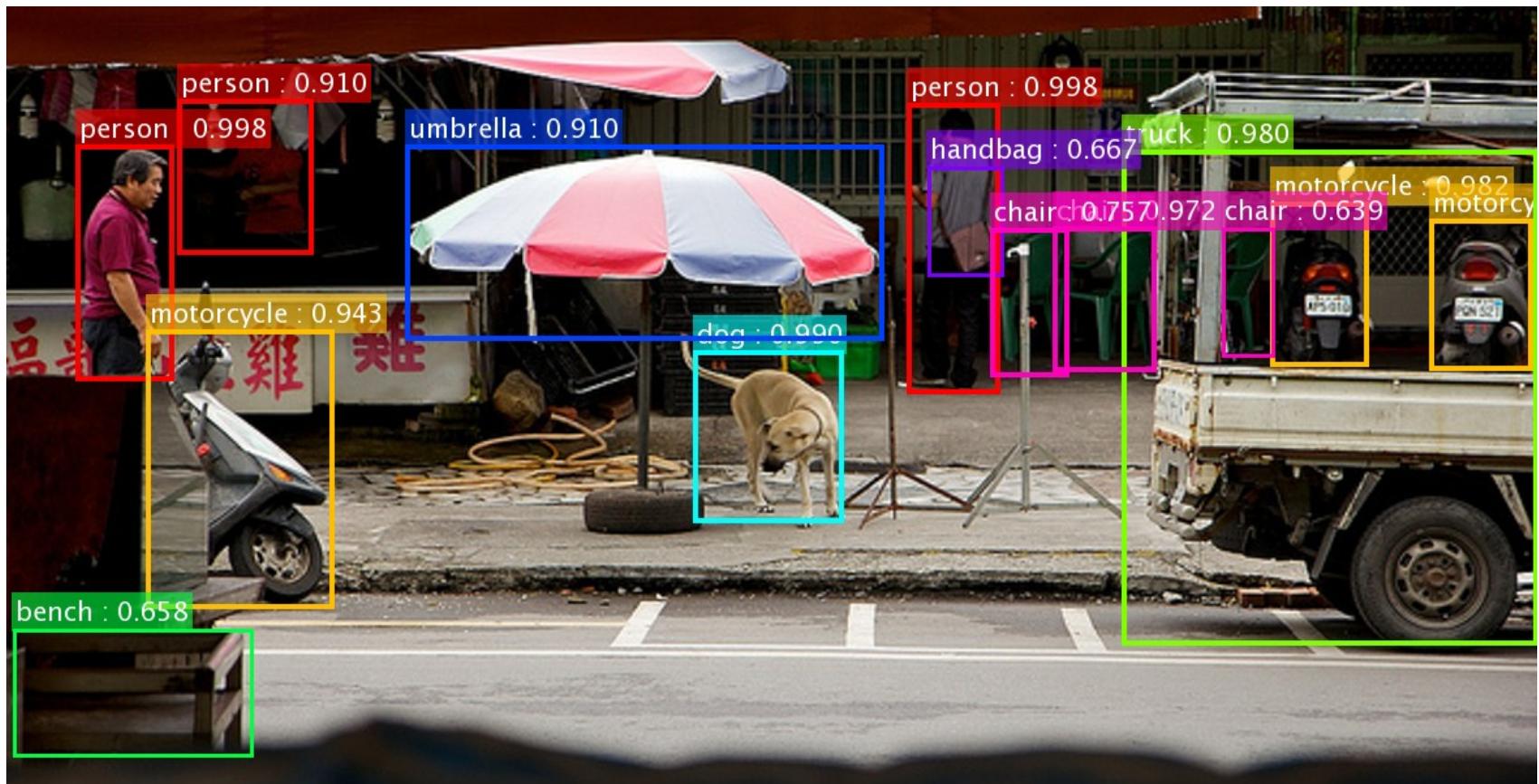
# Object Detection

- RPN **learns** proposals by extremely deep nets
  - We use **only 300 proposals** (no hand-designed proposals)
- Add components:
  - Iterative localization
  - Context modeling
  - Multi-scale testing
- All are based on CNN features; all are end-to-end
- All benefit **more** from **deeper** features - cumulative gains!



## ResNet's object detection result on COCO





Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.  
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.



this video is available online: <https://youtu.be/WZmSMkK9VuA>

Results on real video. Models trained on MS COCO (80 categories).  
(frame-by-frame; no temporal processing)

# More Visual Recognition

## Tasks

ResNet-based methods lead on these benchmarks (incomplete list):

- ImageNet classification, detection, localization
- MS COCO detection, segmentation
- PASCAL VOC detection, segmentation
- Human pose estimation [Newell et al 2016]
- Depth estimation [Laina et al 2016]
- Segment proposal [Pinheiro et al 2016]
- ...

	mean	aero	plane	bicycle	bird	boat	bottle	bus	car	cat	chair	couch	motorcycle	sofa	train	tv
▶ DeepLabv2-CRF [?]	79.7	92.6	60.4	91.6	63.4	76.3	95.0	88.4	90.1	89.5	93.5	89.8	93.5	93.5	93.5	93.5
▶ CASIA_SegResNet_CRF_COCO [?]	79.3	93.8	74.1	91.4	63.1	74.1	95.5	92.1	95.5	89.8	93.5	91.4	93.5	93.5	93.5	93.5
▶ Adelaide_VeryDeep_FCN_VOC [?]	79.1	91.9	48.1	93.4	69.3	75.5	94.2	87.5	91.9	89.8	93.5	91.4	93.5	93.5	93.5	93.5
▶ LRR_4X_COCO [?]	78.7	93.2	49.2	89.4	63.9	74.9	93.5	87.0	91.4	89.8	93.5	91.4	93.5	93.5	93.5	93.5
▶ CASIA_IVA_OASeg [?]	78.3	93.8	41.9	89.4	67.5	71.5	94.6	85.3	89.8	88.4	93.5	91.4	93.5	93.5	93.5	93.5
▶ Oxford_TVGV_HO_CRF [?]	77.9	92.5	59.1	90.3	70.6	74.4	92.4	84.1	88.0	86.3	93.5	91.4	93.5	93.5	93.5	93.5
▶ Adelaide_Context_CNN_CRF_COCO [?]	77.8	92.9	39.6	84.0	67.9	75.3	92.7	83.8	89.8	88.4	93.5	91.4	93.5	93.5	93.5	93.5

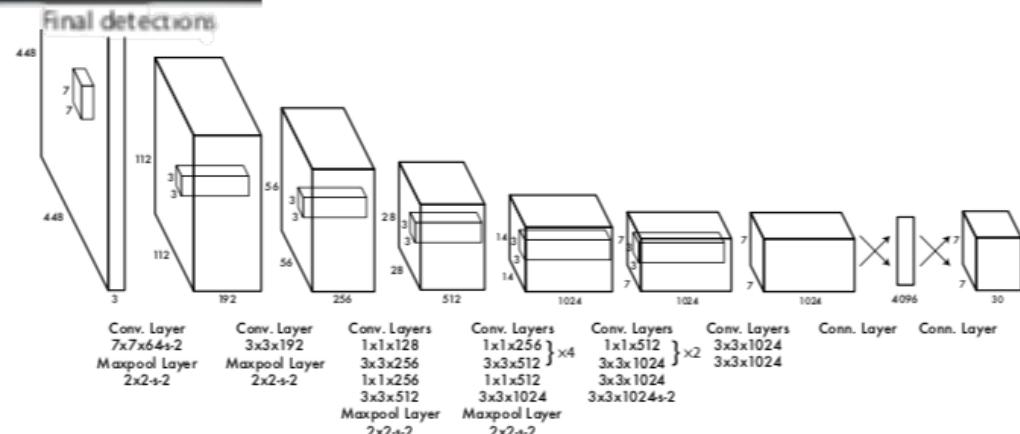
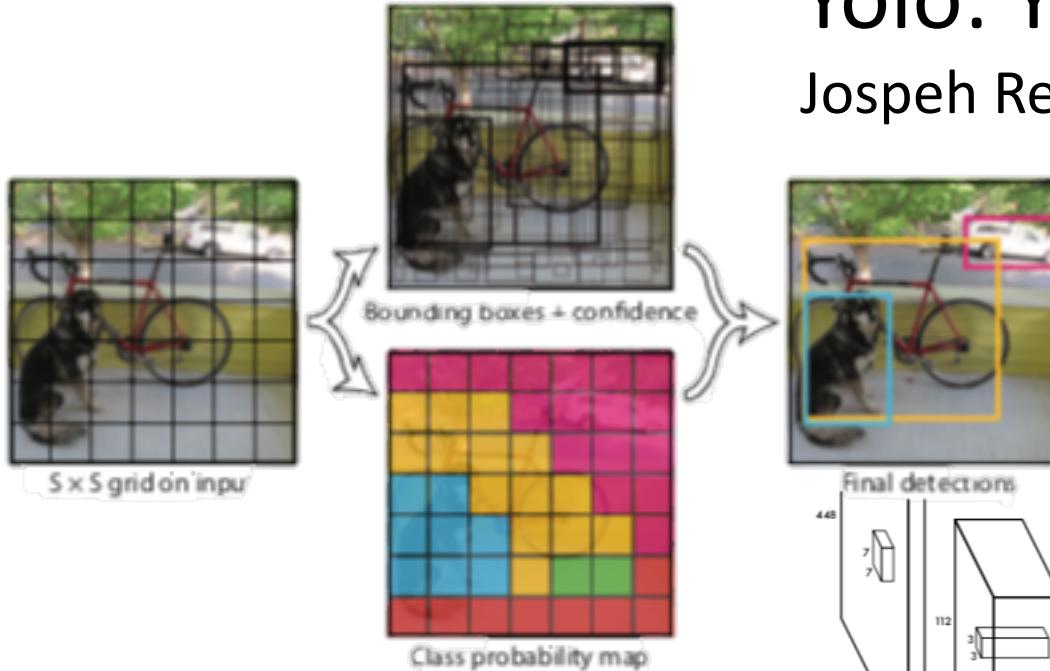
PASCAL segmentation leaderboard

	mean	aero	plane	bicycle	bird	boat	bottle	bus	car	cat	chair	couch	motorcycle	sofa	train	tv
▶ Faster RCNN, ResNet (VOC+COCO) [?]	83.8	92.1	88.4	84.1	75.9	74.1	86.3	87.8	89.8	89.5	93.5	91.4	93.5	93.5	93.5	93.5
▶ R-FCN, ResNet (VOC+COCO) [?]	82.0	89.5	88.3	83.9	74.1	70.7	85.9	84.3	89.2	88.4	93.5	91.4	93.5	93.5	93.5	93.5
▶ DFCN+R-FCN, VGG16, VOC+FLCOCO [?]	80.1	80.1	87.1	73.9	65.0	66.9	80.1	89.8	89.8	89.8	93.5	91.4	93.5	93.5	93.5	93.5
▶ SSD500 VGG16 VOC + COCO [?]	78.7	89.1	85.7	78.9	63.3	57.0	85.3	84.1	92.3	90.0	93.5	91.4	93.5	93.5	93.5	93.5
▶ HFM_VGG16 [?]	77.5	88.8	85.1	76.8	64.8	61.4	85.0	84.1	90.0	88.7	93.5	91.4	93.5	93.5	93.5	93.5
▶ IFRN_07+12 [?]	76.6	87.8	83.9	79.0	64.5	58.9	82.2	82.0	91.4	89.8	93.5	91.4	93.5	93.5	93.5	93.5
▶ ION [?]	76.4	87.5	84.7	76.8	63.8	58.3	82.6	79.0	90.9	89.8	93.5	91.4	93.5	93.5	93.5	93.5

PASCAL detection leaderboard

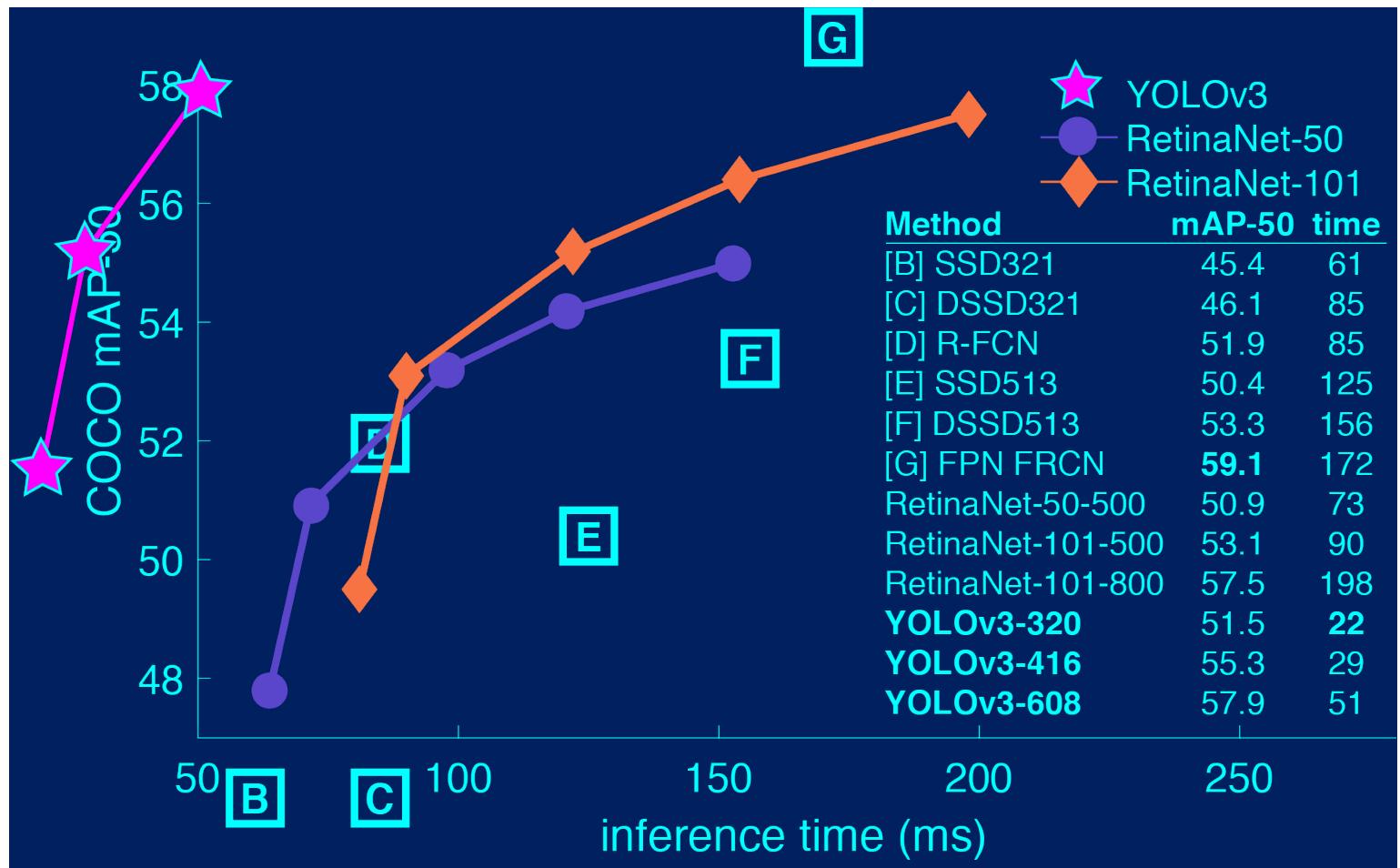
# Yolo: You Only Look Once

Jospeh Redmom et.al ([resumé](#))



- $7 \times 7$  grid of cells
- Each grid cell predicts
  - 2 bounding boxes ( $x, y, w, h$ , confidence) = 10 reals
  - Probabilities over 20 classes
- Final output:  $7 \times 7 \times 30$  tensor ( $30 = 5+5+20$ )
- “the fastest extant object detector” at CVPR 2016

# Yolo 1,2,3...

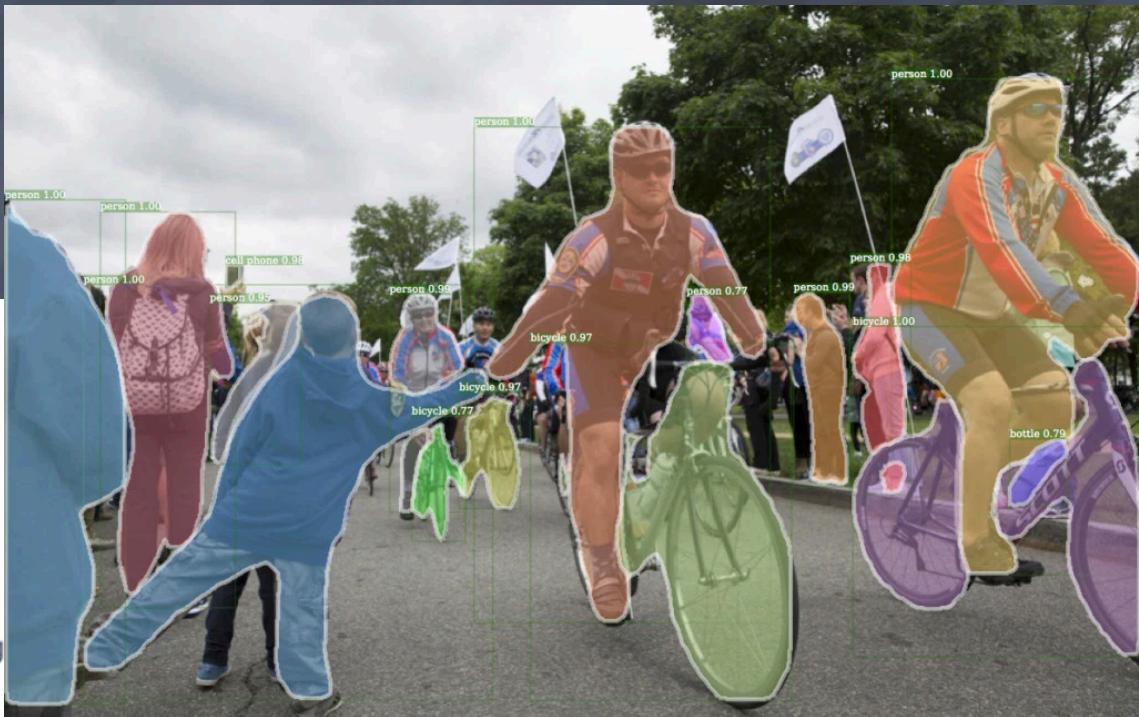


- YOLO: CVPR 2016
- YOLO9000 (YOLOv2) = CVPR 2017
- YOLOv3: Arxiv 2018

Quotes from "[YOLOv3: An Incremental Improvement](#):

Sometimes you just kinda phone it in for a year, you know? ... Spent a lot of time on Twitter. Played around with GANs a little.

# Detectron



Detectron includes implementations of the following:

- Mask R-CNN – *Marr Prize at ICCV 2017*
- RetinaNet – *Best Student Paper Award at ICCV 2017*
- Faster R-CNN
- RPN
- Fast R-CNN
- R-FCN

using the following backbone network architectures:

- ResNeXt{50,101,152}
- ResNet{50,101,152}
- Feature Pyramid Networks (with ResNet/ResNeXt)
- VGG16